

**UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E
ELETRÔNICA**

Thiago Roberto Goularte

**CENTRAL DE MONITORAMENTO PARA BOMBAS DE
INFUSÃO**

Florianópolis

2018

Thiago Roberto Goularte

**CENTRAL DE MONITORAMENTO PARA BOMBAS DE
INFUSÃO**

Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia Elétrica e Eletrônica para a obtenção do Grau de Bacharel em Engenharia Eletrônica.

Orientador: Prof. Walter Pereira Carpes Jr., Dr. Eng.

Florianópolis

2018

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Goularte, Thiago Roberto
Central de monitoramento para Bombas de Infusão
/ Thiago Roberto Goularte ; orientador, Walter
Pereira Carpes Júnior, 2018.
68 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Centro
Tecnológico, Graduação em Engenharia Eletrônica,
Florianópolis, 2018.

Inclui referências.

1. Engenharia Eletrônica. 2. Bomba de Infusão. 3.
Visual Studio. 4. Central de Monitoramento. 5.
ESP32. I. Carpes Júnior, Walter Pereira. II.
Universidade Federal de Santa Catarina. Graduação em
Engenharia Eletrônica. III. Título.

Thiago Roberto Goularte

CENTRAL DE MONITORAMENTO PARA BOMBAS DE INFUSÃO

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de “Bacharel em Engenharia Eletrônica”, e aprovado em sua forma final pelo Departamento de Engenharia Elétrica e Eletrônica.

Florianópolis, 06 de Julho 2018.

Prof. Jefferson Luiz Brum Marques, Dr. Eng.
Coordenador do Curso

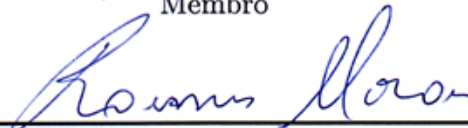
Banca Examinadora:



Prof. Walter Pereira Carpes Jr., Dr. Eng.
Presidente da Banca - Orientador



Prof. Jefferson Luiz Brum Marques, Dr. Eng.
Membro



Prof. Raimes Moraes, Dr. Eng.
Membro

AGRADECIMENTOS

À minha mãe, por todos os sacrifícios feitos e por todo o apoio dado para que eu concluísse minha graduação. Sem você, mãe, eu não teria chegado tão longe

À minha noiva Julliane, por sempre acreditar em mim, mesmo quando nem eu acreditava. Esta conquista é nossa.

À minha irmã por toda a força que tem me dado em todos esses anos.

Ao Walter Carpes, meu orientador e melhor professor que tive na graduação, pela capacidade ímpar de lecionar e, principalmente, motivar à todos.

À meus colegas de graduação, pelo companheirismo e toda a ajuda. Um agradecimento especial à Cristina e Vinícius, que me acompanharam de perto e tornaram meus dias mais agradáveis na universidade.

À todos os meus colegas de trabalho na Hominum que, de alguma forma, me ajudaram neste final de graduação.

À empresa Samtronic por ter acreditado no meu projeto e ter me ajudado a completa-lo.

Try not. Do or do not. There is no try.

Master Yoda

RESUMO

As bombas de infusão têm como objetivo principal administrar fármacos de forma precisa e constante. Mesmo assim, no processo de infusão de fármacos existem anomalias que podem prejudicar o paciente. Entretanto, a grande maioria dos modelos de bomba de infusão disponíveis no mercado estão preparados para detectá-las.

Uma de suas limitações, porém, é o fato de exigirem monitoramento constante junto ao paciente, fazendo com que a equipe de enfermeiros tenha que se deslocar entre os leitos para a verificação manual, tomando um tempo muito grande.

O projeto propõe o desenvolvimento de um sistema de monitoramento sem fio para a bomba de infusão ST 550T2, fornecida gentilmente pela fabricante Samtronic. Primeiramente, os sinais de controle da bomba de infusão são lidos pelo microcontrolador ESP32 e, em seguida, são processados e enviados ao banco de dados MySQL.

O software da central de monitoramento foi desenvolvido no Visual Studio utilizando a plataforma .NET Framework com a linguagem C#. Por fim, foi desenvolvida uma placa de circuito impresso que realiza a interface de conexão entre a bomba o ESP32 de forma que seu encaixe é simples e pouco invasivo para a bomba de infusão.

Palavras-chave: Bomba de infusão. Engenharia biomédica. Central de monitoramento. ESP32. Visual Studio. C++. C#

ABSTRACT

The main purpose of infusion pumps is to administer drugs precisely and consistently. Even so, in the process of drug infusion there are anomalies that can harm the patient, which the vast majority of models of infusion pump available in the market are prepared to detect them. One of their limitations, however, is that they require constant monitoring along with the patient, making the nurses team to move between beds for manual checking, taking a very long time.

The project proposes the development of a wireless monitoring system for the ST 550T2 infusion pump, kindly provided by the samtronic manufacturer.

First, the infusion pump control signals are read by the ESP32 microcontroller and then processed and sent to the MySQL database.

The central monitoring software was developed in Visual Studio using the .NET Framework platform with the C# language.

Finally, a printed circuit board has been developed to make the connection interface between the pump and the ESP32, in a way that its fitting is simple and non-invasive for the infusion pump.

Keywords: Infusion pump. Biomedical Engineering. Center monitor. ESP32. Visual Studio. C++. C#

LISTA DE FIGURAS

| | | |
|-----------|--|----|
| Figura 1 | Estrutura do projeto proposto. | 25 |
| Figura 2 | Bomba de Infusão modelo ST 550T2. | 27 |
| Figura 3 | Painel frontal da bomba de infusão. | 28 |
| Figura 4 | Bomba de infusão aberta, com seu hardware exposto. ... | 29 |
| Figura 5 | Placa traseira da bomba de infusão. | 30 |
| Figura 6 | Placa frontal da bomba de infusão. | 30 |
| Figura 7 | Diagrama esquemático dos leds indicadores visuais presentes na bomba de infusão. | 31 |
| Figura 8 | Pinagem e diagrama esquemático do display de sete segmentos retirados do datasheet disponibilizado pela fabricante. | 33 |
| Figura 9 | Diagrama esquemático do circuito do display de sete segmentos. | 34 |
| Figura 10 | Diagrama esquemático do circuito do barramento de 34 pinos entre a placa frontal e traseira. | 35 |
| Figura 11 | Placa de desenvolvimento ESP32. | 38 |
| Figura 12 | Simulação do divisor de tensão com entrada de 5 V para aproximadamente 3,3 V. | 39 |
| Figura 13 | Placa perfurada montada com todos os resistores soldados. | 39 |
| Figura 14 | Montagem do divisor de tensão com um buffer, isolando a saída da bomba e a entrada do ESP32. | 40 |
| Figura 15 | Sinal S8 da bomba de infusão na entrada do buffer (CH1) e na sua saída (CH2). | 41 |
| Figura 16 | Segunda placa criada para a interface entre bomba e ESP32. | 41 |
| Figura 17 | Circuito completo utilizado para realizar a interface entre a bomba de infusão e o ESP32. | 42 |
| Figura 18 | Conversor de nível lógico bidirecional. | 43 |
| Figura 19 | ESP32 e bomba de infusão conectados por meio de fios. | 43 |
| Figura 20 | Fluxograma do processamento do sinal S4. | 45 |
| Figura 21 | Formas de onda dos sinais de controle dos displays de sete segmentos. | 47 |
| Figura 22 | Formas de onda dos sinais de controle dos displays de sete segmentos depois de iniciar uma infusão. | 47 |

| | |
|--|----|
| Figura 23 Fluxograma do processamento do sinal do conjunto inferior de displays com a temporização correta. | 48 |
| Figura 24 Esquema de segmentos do display | 49 |
| Figura 25 Tela inicial do MySQL Workbench do gerenciador de banco de dados MySQL. | 52 |
| Figura 26 Tabela “led_status” com suas respectivas colunas. | 53 |
| Figura 27 Tabela “display_status” com suas respectivas colunas. . | 53 |
| Figura 28 Windows Visual Studio. | 55 |
| Figura 29 Arquitetura do .NET Framework..... | 56 |
| Figura 30 Tela principal do ambiente de desenvolvimento do Visual Studio. | 57 |
| Figura 31 Interface gráfica da aplicação criada..... | 57 |
| Figura 32 Sequência de execução e parada de infusão..... | 58 |
| Figura 33 Menus do software desenvolvido..... | 58 |
| Figura 34 Layout da placa de circuito impresso..... | 59 |
| Figura 35 Trilhas de cobre da placa de circuito impresso..... | 60 |
| Figura 36 Parte frontal da placa..... | 60 |
| Figura 37 Bomba de infusão aberta com a placa conectada. | 60 |

LISTA DE TABELAS

Tabela 1 Decodificação os sinais dos displays de sete segmentos. . 50

LISTA DE ABREVIATURAS E SIGLAS

| | | |
|-------|--|----|
| ACP | Analgésia Controlada pelo Paciente | 23 |
| Wi-Fi | Wireless Fidelity | 24 |
| NI-MH | Níquel-Cádmio..... | 28 |
| LED | Light Emitting Diode | 29 |
| IDE | Integrated development environment | 37 |
| CH1 | Canal 1..... | 41 |
| CH2 | Canal 2..... | 41 |
| SQL | Structured Query Language | 51 |
| HTML | HyperText Markup Language..... | 52 |
| PHP | Hypertext Preprocessor | 52 |
| VB | Visual Basic | 55 |

LISTA DE SÍMBOLOS

| | | |
|----------|---------------------------|----|
| Ω | Resistência elétrica..... | 38 |
| V | Tensão..... | 38 |
| I | Corrente Elétrica..... | 38 |

SUMÁRIO

| | |
|--|----|
| 1 INTRODUÇÃO | 23 |
| 1.1 OBJETIVO GERAL | 24 |
| 1.2 OBJETIVOS ESPECÍFICOS | 24 |
| 1.3 ESTRUTURA DO PROJETO | 24 |
| 2 ANÁLISE DA BOMBA DE INFUSÃO ST 500T2 | 27 |
| 2.1 FUNCIONAMENTO DA ST 550T2 | 28 |
| 2.2 ANÁLISE DO CIRCUITO DA BOMBA DE INFUSÃO | 29 |
| 2.2.1 Análise do bloco dos leds | 30 |
| 2.2.2 Análise do bloco do display | 32 |
| 2.2.3 Análise do barramento de sinais | 33 |
| 3 INTERFACE ENTRE BOMBA E ESP32 | 37 |
| 3.1 RECEBENDO OS SINAIS COM O ESP32 | 38 |
| 3.1.1 Utilizando Divisor de Tensão | 38 |
| 3.1.2 Utilizando Buffer | 39 |
| 3.2 ESCRITA DOS CÓDIGOS | 44 |
| 3.2.1 Processamento dos Sinais dos Leds | 44 |
| 3.2.2 Processamento dos Sinais dos Displays | 46 |
| 3.2.2.1 Decodificador | 49 |
| 4 INTEGRAÇÃO ENTRE ESP32 E MYSQL | 51 |
| 4.1 CRIAÇÃO DO BANCO DE DADOS NO MYSQL | 51 |
| 4.2 COMUNICAÇÃO ENTRE ESP32 E MYSQL | 52 |
| 5 CRIAÇÃO DO SOFTWARE PARA WINDOWS | 55 |
| 5.1 VISUAL STUDIO E .NET FRAMEWORK | 55 |
| 5.2 DESENVOLVIMENTO DA APLICAÇÃO | 56 |
| 6 PROTÓTIPO FINAL | 59 |
| 7 CONSIDERAÇÕES FINAIS | 61 |
| REFERÊNCIAS | 63 |

1 INTRODUÇÃO

A bomba de infusão é um equipamento médico-hospitalar que permite a administração de um fármaco ou nutriente nas vias venosa, arterial ou esofágica de forma confiável, com controle de fluxo e volume.

Existem vários tipos de bomba de infusão e cada uma com funções específicas, como a bomba de insulina, bomba de ACP, bomba de seringa, bomba peristáltica linear e bomba peristáltica rotativa.

Mesmo com uma excelente precisão e confiabilidade na administração de fármacos, falhas operacionais, nos equipos de infusão ou no equipamento podem ocorrer. Por esse motivo, a grande maioria das bombas de infusão possuem alarmes indicadores de anomalias, como ar na linha do equipo, oclusão na linha, vazão livre, etc. Desta forma, ao detectar alguma anomalia, a infusão é interrompida e um alarme visual e sonoro é acionado.

A limitação, porém, é o fato de a grande maioria das bombas de infusão funcionarem de forma isolada, sem interconectividade com nenhum outro equipamento. Quando um problema é detectado durante a infusão, a presença de um profissional de saúde se faz necessária no local, o qual nem sempre está por perto.

Desta forma, o profissional de saúde deve procurar o leito seguindo apenas a referência sonora que a bomba de infusão emite. Em hospitais com diversos leitos e unidades, esse monitoramento se torna ainda mais complicado e lento.

Hoje, no mercado, existem sistemas de monitoramento para monitores multiparamétricos, que são equipamentos médico-hospitalares responsáveis por monitorar diversos sinais vitais do paciente, como pressão sanguínea, temperatura, oxigenação do sangue, eletrocardiograma, etc. Tais sistemas realizam centralização de informações em que, em um dispositivo alocado na unidade da enfermaria, todos os monitores multiparamétricos da unidade são mostrados ao mesmo tempo. Desta forma, ao perceber alguma anomalia em algum leito, a equipe de enfermaria pode se deslocar diretamente ao local. Tais sistemas são comercialmente chamados de central de monitoramento.

Utilizando as centrais de monitoramento de monitores multiparamétricos como inspiração, teve-se a ideia de desenvolver um sistema parecido para as bombas de infusão. Desta forma, a equipe de enfermaria pode fazer o monitoramento remoto de todas as bombas de infusão na unidade, obtendo assim uma maior agilidade e confiabilidade nos procedimentos de infusão de fármacos.

Em conversas com a empresa Samtronic (fabricante de equipamentos médico-hospitalares, entre eles, bombas de infusão), foi informado o interesse em utilizar uma de suas bombas para a realização do projeto. Desta forma, a empresa disponibilizou uma bomba de infusão peristáltica rotativa modelo ST 550T2.

1.1 OBJETIVO GERAL

Desenvolver um sistema de monitoramento sem fio para a bomba de infusão modelo ST 500T2 com interface simples e objetiva de forma a obter um maior controle nos procedimentos de infusão.

1.2 OBJETIVOS ESPECÍFICOS

Visando a atingir o objetivo principal, alguns objetivos específicos são requeridos, entre eles:

- Obtenção dos sinais responsáveis pelos indicadores visuais da bomba de infusão;
- Obtenção dos sinais dos displays de sete segmentos presentes;
- Processamento e envio das informações via Wi-Fi a um banco de dados;
- Criação de um banco de dados para receber as informações pertinentes;
- Desenvolvimento de uma interface gráfica de forma a mostrar as informações obtidas da bomba de infusão;
- Toda a comunicação deverá ser feita via Wi-Fi, mas sem conexão à internet, por questões de segurança de dados.

1.3 ESTRUTURA DO PROJETO

O projeto desenvolvido possui quatro partes bem definidas:

- Leitura dos sinais da bomba de infusão;

- Processamento e envio dos dados com um microcontrolador. O microcontrolador utilizado foi o ESP32;
- Criação e gerenciamento de um banco de dados com o MySQL;
- Criação de uma interface gráfica da Central de Monitoramento com uma aplicação .NET Framework para Windows.

A estrutura pode ser vista no diagrama na figura 1.

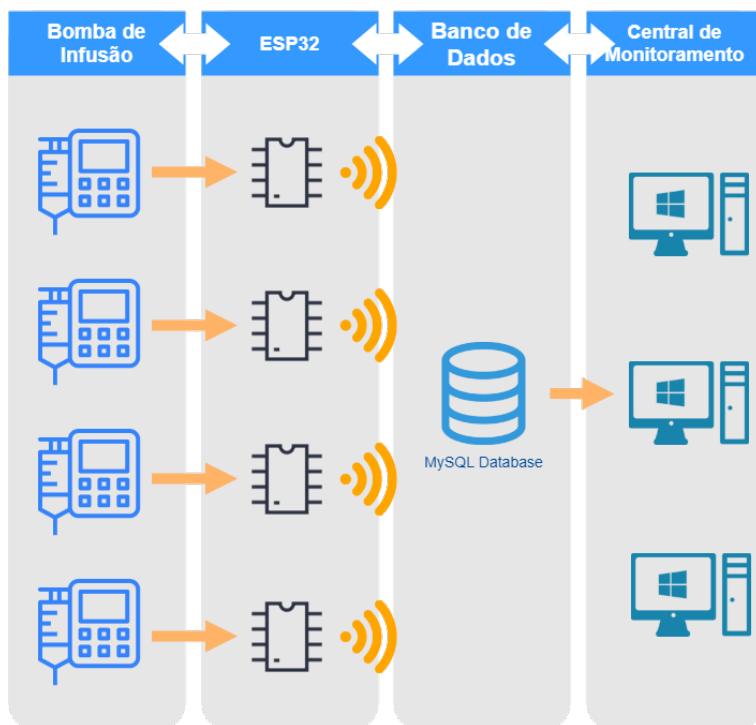


Figura 1 – Estrutura do projeto proposto.

2 ANÁLISE DA BOMBA DE INFUSÃO ST 500T2

A bomba de infusão utilizada no projeto foi a ST 550T2, mostrada na figura 2, fabricada pela empresa Samtronic. Ela é, atualmente, o modelo mais antigo ainda vendido pela fabricante, além de ser o único modelo de bomba peristáltica rotativa disponível.



Figura 2 – Bomba de Infusão modelo ST 550T2.

FONTE: (SAMTRONIC, 2007)

Segundo a fabricante (SAMTRONIC, 2007), algumas de suas especificações são:

- Vazão de até 999,9 ml/h, com incremento de 0,1 ml/h;
- Controle de volume a infundir de até 9999,9 ml;
- Limite de tempo de infusão de 1 min até 99 h 59 min;
- Possui sete células de bateria NI-MH com 2100 mAh de capacidade, concedendo autonomia de até 4 h;
- Sistema mecânico de alívio de pressão de oclusão impedindo uma sobreinfusão acidental;
- Detecção de ar na linha, com possibilidade de desligamento somente para administração de terapia enteral;
- Detecção de oclusão na linha;

- Detecção de vazão livre na linha;
- Alarmes sonoros.

2.1 FUNCIONAMENTO DA ST 550T2

Ao analisar o manual do equipamento (SAMTRONIC, 2007), pôde-se encontrar uma descrição completa de todos os indicadores visuais e botões presentes no painel frontal, como mostra a figura 3.

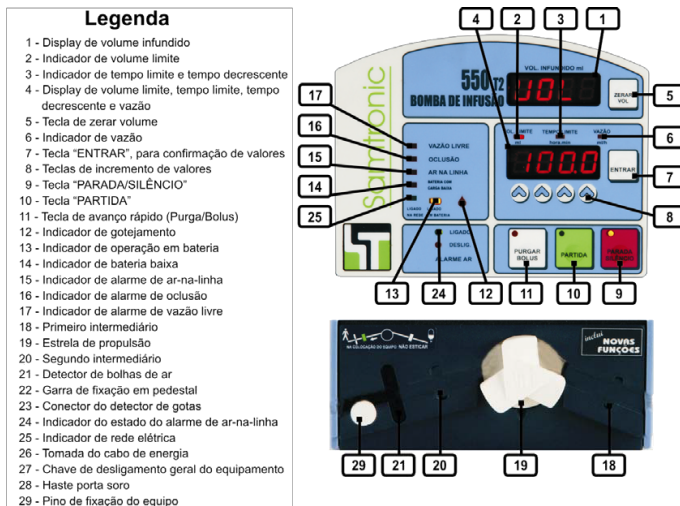


Figura 3 – Painel frontal da bomba de infusão.

FONTE: (SAMTRONIC, 2007)

Quando a bomba de infusão for ligada, suas informações mostradas serão as mesmas encontradas na imagem 3, com exceção do valor mostrado no display.

Ela possui dois modos de configuração: num deles, escolhe-se o volume e o tempo a serem infundidos, e ela realiza o cálculo da vazão. No outro, escolhe-se o volume e a vazão, e ela realiza o cálculo do tempo total.

Desta forma, caso se deseje, por exemplo, escolher o volume e o tempo infundidos, o procedimento de configuração segue:

Escolhe-se o valor de volume a ser infundido pelas teclas de incremento de valores e em seguida deve-se pressionar a tecla “Entrar”.

Ao pressionar a tecla “Entrar”, o led indicador de volume limite se apaga e o indicador de tempo limite acende, indicado que o valor mostrado no display agora pertence ao tempo de infusão desejado. Deve-se, então, escolher o valor de tempo total de infusão e, novamente, pressionar a tecla “Entrar”.

O led indicador de vazão acende, indicando que o valor mostrado é a vazão da bomba, cujo cálculo foi realizado automaticamente. Pressiona-se novamente a tecla “Entrar” e a bomba irá gerar um sinal sonoro.

Neste momento, deve-se pressionar a tecla de “Partida” e, então, o led indicador de parada irá apagar enquanto o de partida irá acender.

Se durante a infusão a bomba detectar ar, oclusão ou vazão livre na linha, a bomba interromperá o procedimento e acionará o respectivo alarme.

2.2 ANÁLISE DO CIRCUITO DA BOMBA DE INFUSÃO

Ao abrir a bomba de infusão, tem-se o acesso a todo o seu hardware, como mostrado na figura 4.

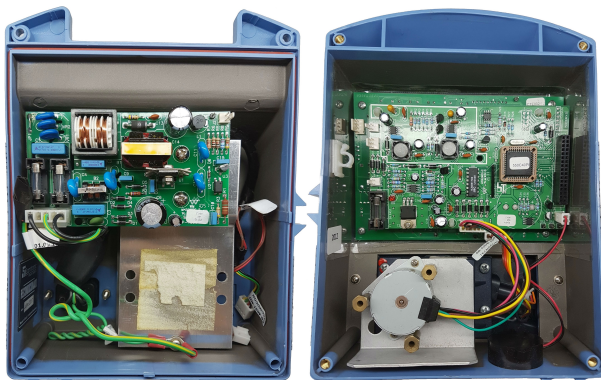


Figura 4 – Bomba de infusão aberta, com seu hardware exposto.

O foco, porém, é analisar o módulo de controle, o qual é constituído de duas placas, como mostrado nas figuras 5 e 6.

A placa frontal da bomba de infusão possui um total de 15 leds

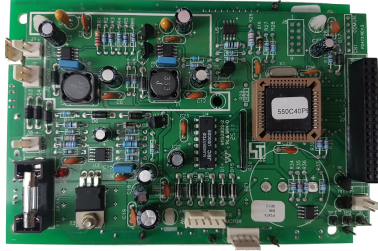


Figura 5 – Placa traseira da bomba de infusão.

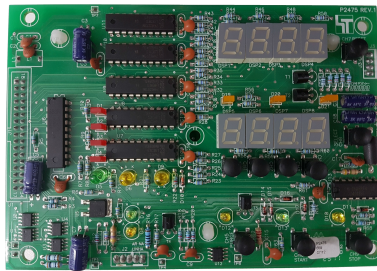


Figura 6 – Placa frontal da bomba de infusão.

indicadores visuais, os quais indicam infusão iniciada, infusão parada, purgar, volume limite selecionado, tempo limite selecionado, vazão selecionada, alarme de vazão livre, alarme de ar na linha, alarme de oclusão, sistema de ar na linha habilitado, sistema de ar na linha desabilitado, detecção de gotejamento, ligado na rede elétrica, ligado em bateria e bateria baixa.

2.2.1 Análise do bloco dos leds

Através de uma inspeção visual, inicialmente imaginou-se que o controle dos leds poderia vir diretamente do microcontrolador U6 (presente na placa traseira). Porém, ao seguir as trilhas e medindo continuidade com um multímetro, percebeu-se que eles são habilitados por dois circuitos integrados “74HC273N”, o qual possui 8 flip-flops tipo D integrados em seu encapsulamento (NEXPERIA, 2016).

Essa análise se confirmou quando se teve acesso ao diagrama

esquemático apresentado na figura 7.

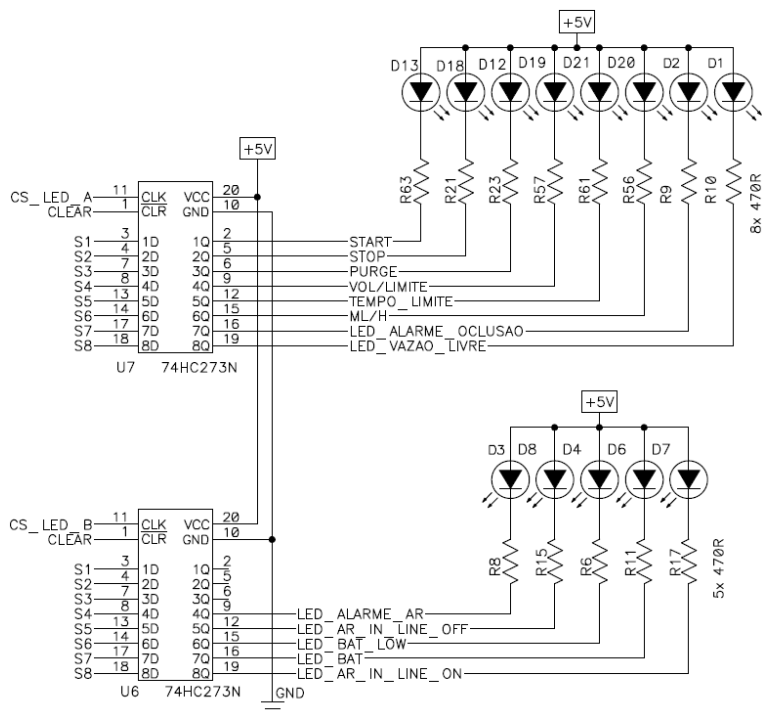


Figura 7 – Diagrama esquemático dos leds indicadores visuais presentes na bomba de infusão.

O status dos leds (D1- vazão livre, D2- oclusão, D20- vazão, D12- purgar, D21- tempo limite, D13- partida, D19- volume limite e D18- parada) é controlado pelo circuito integrado U7, que recebe as informações da linha de dados S1 a S8 e do comando do microcontrolador (na placa traseira).

O status do sistema de ar na linha (D7- ligado, D8- desligado), controle de ar na linha (D3- ar na linha), leds de alimentação (D4- bateria com carga baixa, D6- ligado em bateria) é controlado pelo circuito integrado U6, que recebe as informações da linha de dados S1 a S8 e do comando do microcontrolador (na placa traseira).

O microcontrolador, então, sincroniza os sinais S1 a S8 com os sinais CS_LED_A e CS_LED_B de forma que se habilite um circuito integrado a cada ciclo de relógio e se envie o sinal correto para os leds

(SMITH; CEDRA, 2009).

Esta topologia faz com que uma parte dos leds fique acesa durante um ciclo de relógio e apagada durante outro. Porém, essa operação é realizada centenas de vezes em um segundo, de forma que o olho humano não consegue percebê-la.

Se, por exemplo, a vazão estiver selecionada, o led de vazão (D20) deverá acender. Verificando no circuito esquemático da figura 7, o sinal S6 é enviado para o pino 6D de ambos os circuitos integrados U6 e U7. Porém, na saída do pino 6D do circuito integrado U6, o led de vazão é acionado, enquanto na saída do pino 6D do circuito integrado U7, o led de bateria baixa é acionado.

Desta forma, para acender o led de vazão, o CS_LED_A deverá ficar em nível lógico baixo, enquanto o CS_LED_B deve ficar em nível lógico alto. Assim, o sinal S6 será habilitado apenas na saída do circuito integrado U7.

É interessante notar que a lógica de acionamento dos leds é invertida da comumente usada, pois como os leds possuem seus anodos em comum, para criar uma diferença de potencial no led, os flip-flops não devem aplicar tensão alguma.

2.2.2 Análise do bloco do display

O aparelho possui dois conjuntos de painéis para visualização de dados. Cada um destes painéis possui quatro displays de sete segmentos do modelo anodo comum “XDMR08A” da fabricante SunLED (SUNLED, 2014). Analisando o datasheet da fabricante, tem-se o diagrama esquemático mostrado na figura 8.

Cada display de sete segmentos pode ser visto como 8 leds com seus anodos ligados em +Vcc, exatamente da mesma forma que foi verificada no bloco dos leds. Cada caractere formado corresponde a uma sequência de leds acesos e apagados. A forma mais simples de decodificá-lo seria pegando cada um dos pinos de cada um dos displays e ligá-los a um pino de entrada no microcontrolador. Porém, para isso seriam necessários 8 pinos para cada display, totalizando 64 pinos de entrada apenas para essa finalidade, tornando esta solução inviável.

Da mesma forma que no bloco de leds, os displays de sete segmentos também são controlados por flip-flops “74HC273N”, os quais possuem exatamente 8 pinos de saída: um para cada pino do display.

Esta análise se confirmou quando se teve acesso ao diagrama esquemático apresentado na figura 9.

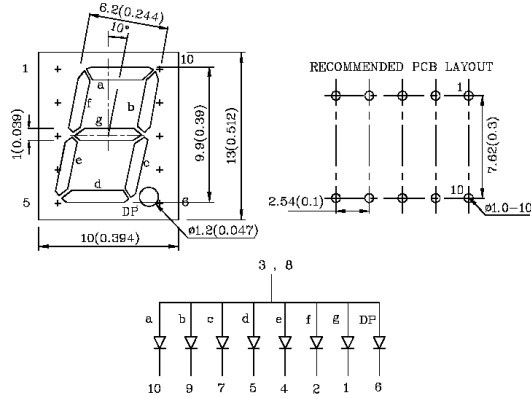


Figura 8 – Pinagem e diagrama esquemático do display de sete segmentos retirados do datasheet disponibilizado pela fabricante.

FONTE:(SUNLED, 2014)

Após a análise do esquemático, pode-se verificar que os circuitos integrados (flip-flops) U8, U9, U10, U11, os quais recebem os sinais S1 a S8, são habilitados por CLKM, CLKC, CLKD e CLKU e, posteriormente, enviam informações para os dois conjuntos de displays. Porém, um sinal de controle chamado de DIGIVOL habilita o conjunto formado por DSP5, DSP6, DSP7 e DSP8, e outro sinal de controle, chamado DIGIVAZ, habilita o conjunto formado por DSP1, DSP2, DSP3 e DSP4.

Desta forma, quando se quer atualizar o display DSP5, por exemplo, o sinal CLKM habilita o flip-flop U11 de forma a mostrar na saída o valor de S1, S2, S3, S4, S5, S6, S7 e S8 armazenado e os sinais CLKC, CLKD, CLKU desabilitam os flip-flops U8, U9 e U10. Ao mesmo tempo, o sinal de controle DIGIVAZ desabilita o conjunto superior, enquanto o sinal de controle DIGIVOL habilita o conjunto inferior, fazendo com que apenas o display DSP5 seja atualizado. Essa lógica pode ser utilizada quando se quer atualizar qualquer um dos oito displays.

2.2.3 Análise do barramento de sinais

A última análise pertinente para o projeto é referente a onde, fisicamente, será feita a conexão entre a bomba de infusão e o sistema de

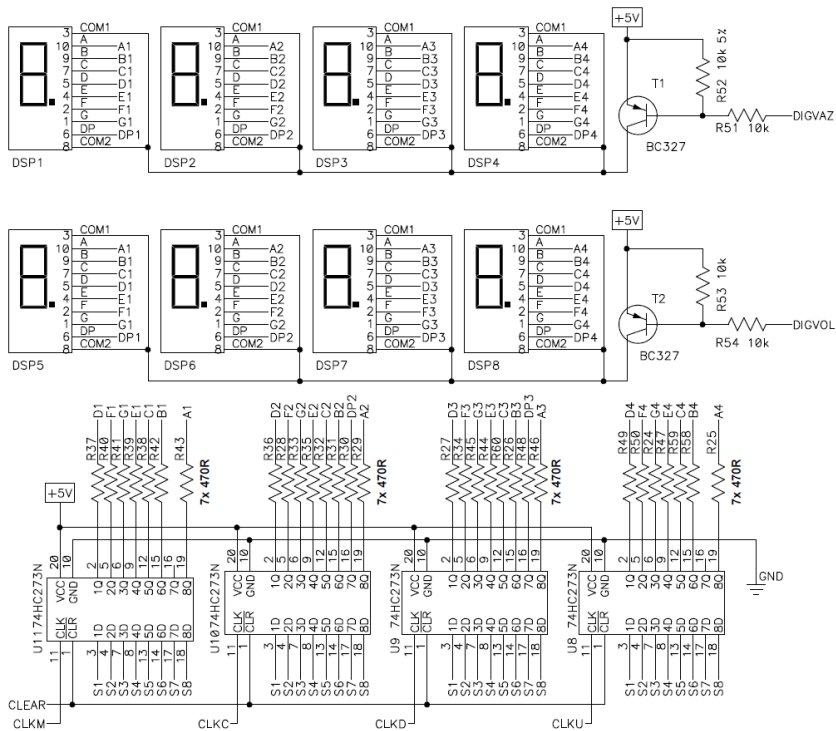


Figura 9 – Diagrama esquemático do circuito do display de sete segmentos.

comunicação que será utilizado. Conforme dito anteriormente, coletar os sinais da bomba diretamente nos pinos dos leds ou dos displays é completamente inviável, pois seriam necessárias 15 entradas para os leds e mais 64 entradas para os displays de sete segmentos. Outro agravante é em relação à quantidade de fios que seriam soldados nos mais diversos pontos da placa.

Analisando os esquemáticos da bomba de infusão, nota-se que existem no total 15 sinais de interesse que, em teoria, possibilitam obter todas as informações mostradas. E são estes:

- CS_LED_B - Sinal de controle do circuito integrado U6;
- CS_LED_A - Sinal de controle do circuito integrado U7;
- CLKU - Sinal de controle do circuito integrado U8;

- CLKD - Sinal de controle do circuito integrado U9;
- CLKC - Sinal de controle do circuito integrado U10;
- CLKM - Sinal de controle do circuito integrado U11;
- S1 - Informação vinda do microcontrolador;
- S2 - Informação vinda do microcontrolador;
- S3 - Informação vinda do microcontrolador;
- S4 - Informação vinda do microcontrolador;
- S5 - Informação vinda do microcontrolador;
- S6 - Informação vinda do microcontrolador;
- S7 - Informação vinda do microcontrolador;
- S8 - Informação vinda do microcontrolador;
- DIGIVOL - Sinal que habilita conjunto inferior de displays.

A placa frontal da bomba de infusão é responsável por realizar o controle (por meio de botões) e pela interface visual, enquanto que o processamento é realizado na parte traseira. Assim, verificou-se que existe apenas um barramento de 34 pinos que realiza a comunicação entre as duas placas.

Soldando um barra-pinos fêmea por cima do barramento na placa traseira, tem-se o ponto ideal para realizar a interface entre a bomba de infusão e o sistema de comunicação, conforme mostrado na figura 10.

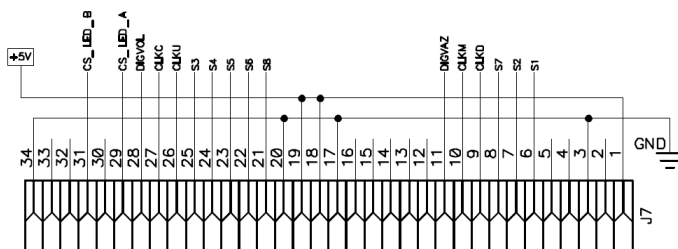


Figura 10 – Diagrama esquemático do circuito do barramento de 34 pinos entre a placa frontal e traseira.

3 INTERFACE ENTRE BOMBA E ESP32

O sistema responsável pela comunicação com bomba de infusão será uma placa de desenvolvimento chamada ESP32 fabricada pela empresa Expressif, mostrada na figura 11. Algumas de suas principais características (ESPRESSIF, 2018) são:

- Processador principal: LX6 32-bit Dual-core, operando em 2-240 MHz;
- Processador secundário: ULP (Ultra Low Power coprocessor);
- FLASH: 4MB;
- RAM: 520 kB;
- GPIO: 34, com 3,3 V e 12 mA;
- ADC: 18, com resolução de 12 bits;
- DAC: 2, com resolução 8 bits;
- Wi-Fi: 2,4 GHz, 802.11 b/g/n;
- Bluetooth: Bluetooth Low Energy v4.2 (BLE);
- Acelerador via hardware para encriptações, hash e afins. (AES, RSA, SHA e ECC);
- True Random Number Generator (TRGN);
- 4 Timers de 64 bits;
- 4 Watchdogs;
- 10 Sensores de Touch Capacitivo;
- 1 Sensor de temperatura interno;
- 1 Sensor de efeito Hall.

Atualmente, existem alguns Ambientes de Desenvolvimento de aplicações para o ESP32, como o ESP-IDE, Arduino-IDE, PlatformIO.

Os principais motivos para utilizar o ESP32 no desenvolvimento do projeto foram o fato dele ser compatível com a Arduino-IDE (plataforma de desenvolvimento para a placa Arduino) e possuir Wi-Fi já integrado.

Porém, é válido lembrar que todos os códigos e bibliotecas utilizadas são totalmente compatíveis com qualquer Arduino ou similares.



Figura 11 – Placa de desenvolvimento ESP32.

3.1 RECEBENDO OS SINAIS COM O ESP32

Segundo o datasheet da fabricante (ESPRESSIF, 2018), o ESP32 aceita alimentação máxima de $V_{DD} = 3,6$ V e suporta tensões até $V_{DD} + 0,3$ V nos pinos de entrada, chegando a valores máximos de 3,9 V na entrada.

Uma vez que a bomba de infusão trabalha com nível lógico de 5 V, corre-se o risco de danificar a placa aplicando uma tensão maior do que a originalmente projetada. Desta forma, optou-se uma solução simples: utilizando um divisor de tensão em cada um dos pinos de sinais da bomba de infusão.

3.1.1 Utilizando Divisor de Tensão

O divisor de tensão é uma topologia utilizada para diminuir uma tensão de referência de acordo com os valores de resistência utilizados (SMITH; CEDRA, 2009). Sua topologia é mostrada na figura 12.

Sabendo que a corrente que passa em R1 é igual à corrente que passa em R2, temos:

$$\frac{V_i - V_o}{R_1} = \frac{V_o - 0}{R_2} \quad (3.1)$$

$$\frac{V_i}{R_1} = \frac{V_o}{R_2} + \frac{V_o}{R_1} \quad (3.2)$$

$$\frac{V_i}{R_1} = V_o \frac{(R_2 + R_1)}{(R_2 R_1)} \quad (3.3)$$

$$V_o = V_i \frac{R_2}{(R_2 + R_1)} \quad (3.4)$$

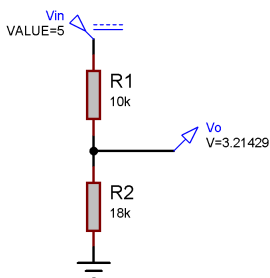


Figura 12 – Simulação do divisor de tensão com entrada de 5 V para aproximadamente 3,3 V.

Para realizar a interface, então, o circuito foi montado em uma placa de cobre universal e os componentes devidamente soldados, como mostrado na figura 13.

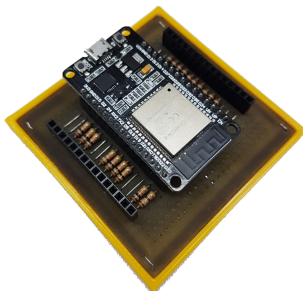


Figura 13 – Placa perfurada montada com todos os resistores soldados.

O grande problema desta topologia na aplicação proposta é o fato de todos os sinais de controle da bomba de infusão ficarem em paralelo com um resistor de carga de 18 k Ω . Isso causa uma falha completa nos leds, displays e até mesmo nos botões de controle da mesma.

3.1.2 Utilizando Buffer

Na primeira tentativa, houve uma falta de atenção em relação à carga que ficaria em paralelo nas saídas dos sinais da bomba de infusão. Uma forma simples de solucionar esse problema é utilizando buf-

fers, aproveitando suas características de alta impedância de entrada, isolando dois estágios.

Escolheu-se então o circuito integrado “SN74LS244”, o qual possui 8 buffers integrados em seu encapsulamento, necessitando de apenas duas unidades para realizar toda a interface de comunicação.

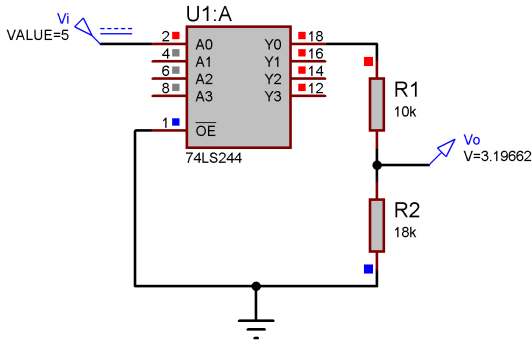


Figura 14 – Montagem do divisor de tensão com um buffer, isolando a saída da bomba e a entrada do ESP32.

Considerando que há um total de 15 pinos de sinais para realizar a leitura, precisa-se então de 15 buffers para tal aplicação.

Em tese, o buffer transfere o valor de tensão na entrada diretamente para a saída, isolando os dois estágios. Porém, na prática, existem as limitações físicas do amplificador operacional.

Ao realizar testes em protoboard com o “SN74LS244”, notou-se uma queda de tensão entre a entrada e a saída, como mostrado na figura 15. Os sinais de controle da bomba de infusão trabalham na faixa dos 5 V e, ao verificar a saída dos buffers, a tensão foi de aproximadamente 3,9 V, valor este dentro da tolerância do ESP32.

Desta forma, não se viu mais a necessidade de utilizar um divisor de tensão, mantendo apenas um resistor de 10 k Ω na saída de cada buffer para limitar a corrente de entrada no ESP32 e, consequentemente, conceder uma maior segurança. Desta forma, montou-se o circuito mostrado no esquemático da figura 17.

Para facilitar os testes e prevendo a necessidade de alteração de pinos de entrada do ESP32, foi utilizada mais uma placa perfurada universal para soldar os componentes em seus devidos lugares, mas ainda assim tendo a liberdade de alterar as entradas e saídas do circuito, como mostrado na figura 16. Posteriormente, pôde-se conectar a bomba

de infusão ao ESP32 com sucesso.

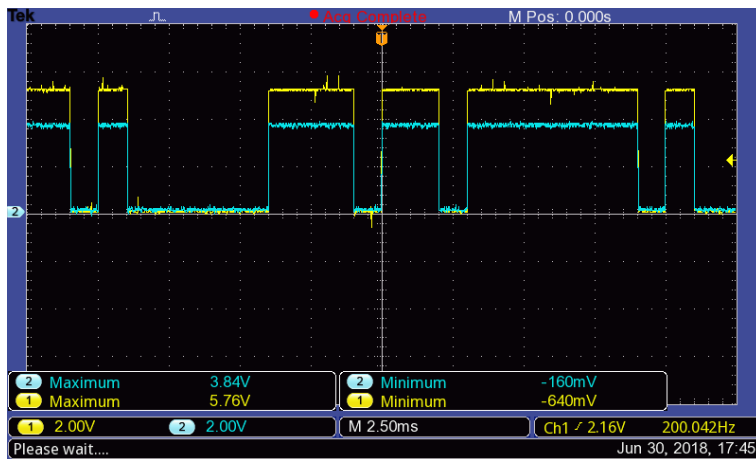


Figura 15 – Sinal S8 da bomba de infusão na entrada do buffer (CH1) e na sua saída (CH2).

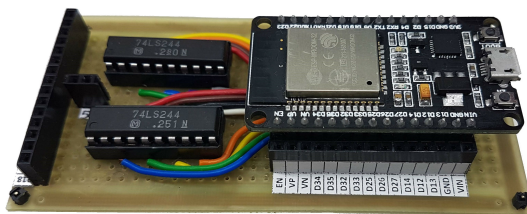


Figura 16 – Segunda placa criada para a interface entre bomba e ESP32.

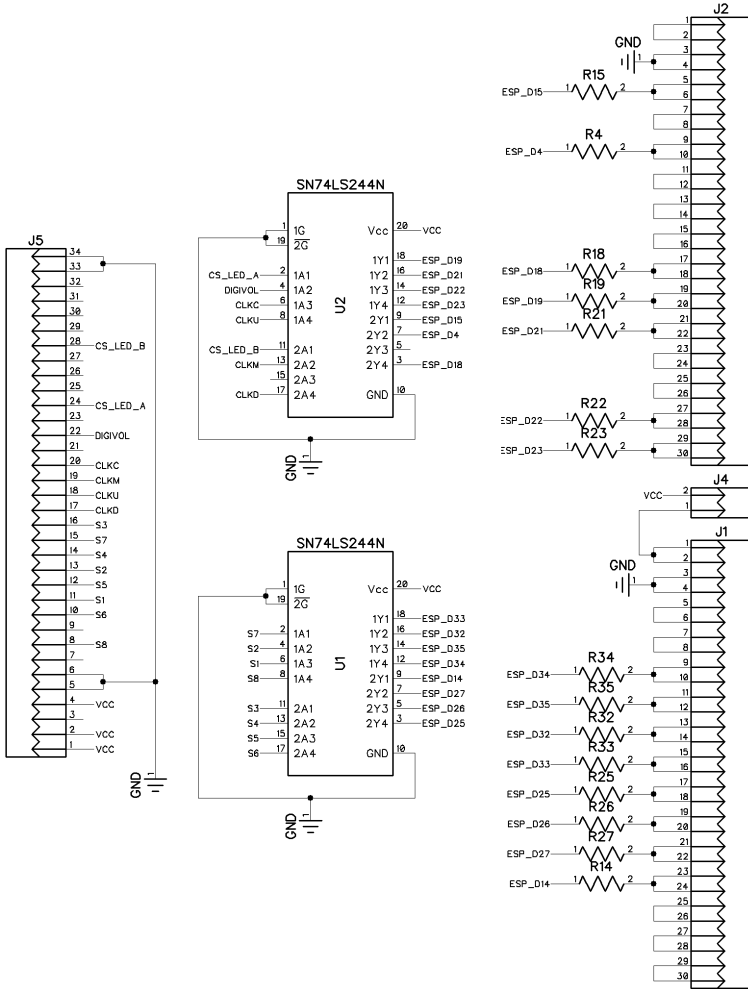


Figura 17 – Circuito completo utilizado para realizar a interface entre a bomba de infusão e o ESP32.

Apesar do buffer utilizado ter funcionado corretamente, ele ainda não é a melhor opção neste caso. Além da necessidade de isolar os dois estágios, ainda deveria-se adicionar o divisor de tensão. Desta forma, a topologia ideal é utilizar os chamados conversores lógicos. São circuitos simples baseados em mosfets e resistores de forma a realizar a conversão

de níveis lógicos de 3,3 V para 5 V e vice-versa, como pode-se verificar na figura 18.

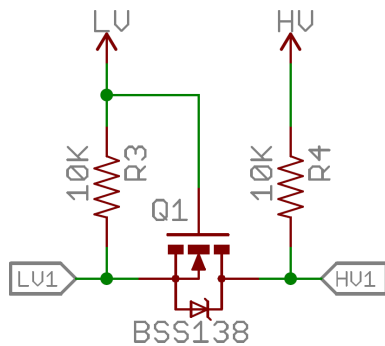


Figura 18 – Conversor de nível lógico bidirecional.
FONTE: (SPARKFUN, 2018)

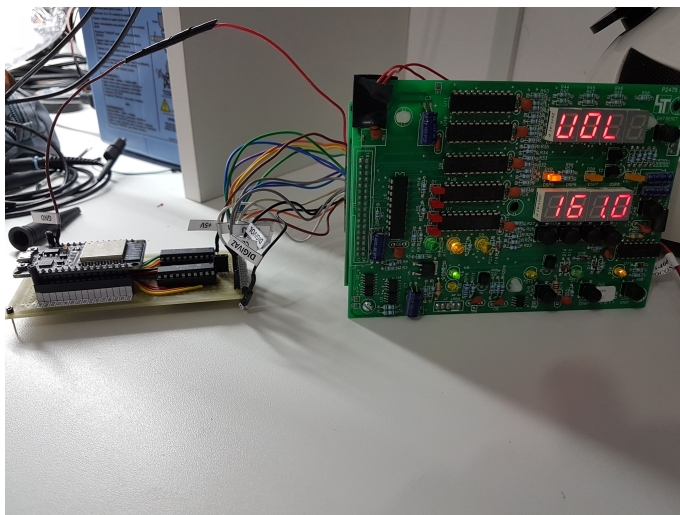


Figura 19 – ESP32 e bomba de infusão conectados por meio de fios.

3.2 ESCRITA DOS CÓDIGOS

Após a criação da interface física entre a bomba de infusão e o ESP32, deve-se programá-lo para ler e processar os sinais recebidos. A programação do ESP32 foi totalmente escrita em C++ no ambiente de desenvolvimento de aplicações Arduino-IDE devido à grande familiaridade com ele.

3.2.1 Processamento dos Sinais dos Leds

Para a leitura dos sinais dos leds, foi criada uma variável inteira para cada um deles que será atualizada a cada ciclo de relógio do ESP32. Neste caso, a única informação pertinente é saber se eles estão ligados ou desligados, indicando o estado atual da bomba de infusão.

As variáveis criadas foram:

- st - Led Start;
- stp - Led Stop;
- pg - Led Purge;
- vlim - Led Volume limite;
- tlim - Led Tempo limite;
- mlh - Led Vazão;
- aoc - Led Alarme de oclusão;
- avl - Led Vazão Livre;
- aal - Led Alarme de ar na linha;
- aloff - Led Ar na linha desligada;
- batl - Led Bateria baixa;
- bat - Led Ligada em bateria;
- alon - Led Ar na linha ligada;
- rede - Led Ligada na rede elétrica.

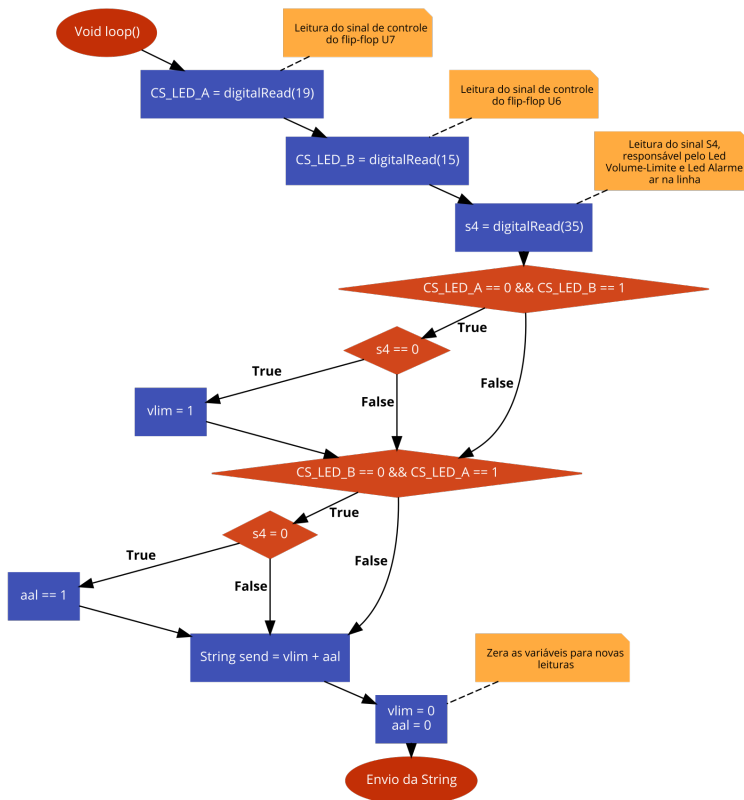


Figura 20 – Fluxograma do processamento do sinal S4.

No fluxograma da figura 20 temos um exemplo de como o processamento do sinal S4 é feito de forma a atualizar o estado dos leds Volume limite e Alarme ar na linha.

Para todos os outros leds da bomba de infusão, a mesma metodologia é utilizada. Ao final da leitura e processamento de todos os sinais, uma string chamada “send” é atualizada. Essa string será utilizada posteriormente no envio das informações ao banco de dados.

3.2.2 Processamento dos Sinais dos Displays

Conforme visto anteriormente, existem dois conjuntos com quatro displays de sete segmentos cada. Cada conjunto é habilitado pelos sinais de controle DIGIVOL e DIGIVAZ. Os displays de sete segmentos, por sua vez, são atualizados pelo barramento de sinais S1 a S8, os quais são habilitados na saída dos flip-flops pelos sinais CLKM, CLKC, CLKD, CLKU.

Apesar de haver mais sinais de controle e ainda haver a necessidade de decodificar os sinais em números e letras, teoricamente, o princípio de funcionamento é o mesmo encontrado nos leds da bomba de infusão. Desta forma, criou-se um código semelhante ao anterior, com a adição de um decodificador dos displays.

Porém, ao rodar o código, o que foi obtido não foram os caracteres mostrados na bomba de infusão, mas sim uma sequência de caracteres. E verificou-se que o sinal de controle DIGIVOL não estava fazendo diferença alguma no código, gerando sempre os mesmos erros. Por exemplo, o sinal decodificado do display DSP5 era, na verdade, correspondente à sequência de caracteres apresentados nos displays DSP1 a DSP8.

Supôs-se, então, que um erro de sincronização entre a leitura dos sinais estava ocorrendo. Decidiu-se analisar as formas de onda destes sinais com um osciloscópio, com o intuito de entender melhor como eles são gerados pela bomba. Os sinais analisados no osciloscópio foram DIGIVOL, DIGIVAZ, CLKM, CLKC, CLKD, CLKU.

O Osciloscópio utilizado foi o TBS 1052B, da fabricante Tektronix, e uma de suas limitações é possuir apenas dois canais. De fato, para entender exatamente como os sinais se comportam, era necessário ler todos os seis sinais de controle ao mesmo tempo. Para contornar esta limitação, salvou-se cada uma das formas de onda separadamente num pendrive e, no software Matlab, plotou-se os seis sinais juntos, conforme pode-se ver na figura 21.

Analisando a imagem 21 pôde-se perceber que depois da borda de subida do sinal DIGIVAZ, há um atraso de 2 ms e o sinal CLKM aparece seguido dos sinais CLKC, CLKD E CLKU atrasados em 1 ms entre si. O mesmo acontece para o sinal DIGIVAZ.

Pôde-se assim deduzir que, quando há a borda de subida do sinal DIGIVOL, o conjunto de displays inferior é desabilitado, e os sinais de controle atualizam o conjunto de displays superior de forma sequencial. E quando há o sinal de borda de SUBIDA do sinal DIGIVAZ, o contrário ocorre.

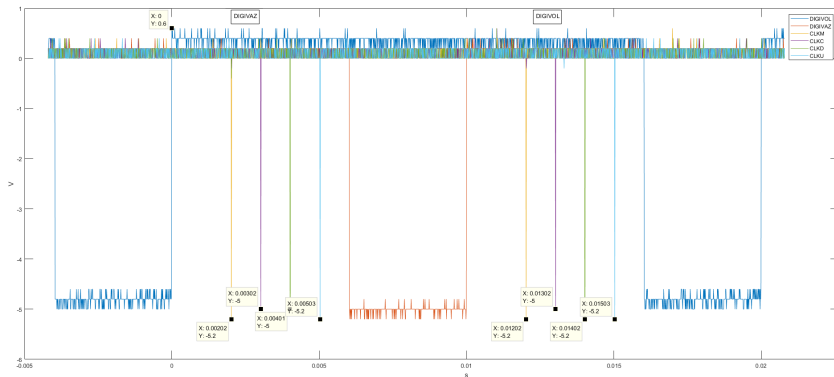


Figura 21 – Formas de onda dos sinais de controle dos displays de sete segmentos.

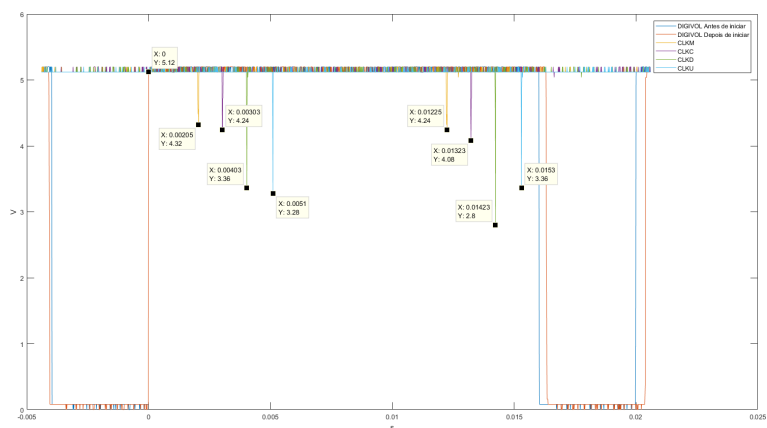


Figura 22 – Formas de onda dos sinais de controle dos displays de sete segmentos depois de iniciar uma infusão.

Basta, então, criar uma interrupção (SANTOS, 2017) no código de forma que, quando há a borda de sinal de subida de DIGIVOL, espera-se o tempo necessário para realizar a leitura dos outros sinais, como mostrado na figura 23.

Um problema encontrado com esta solução é a sensibilidade quanto ao tempo de atraso, pois uma mudança no atraso entre os sinais

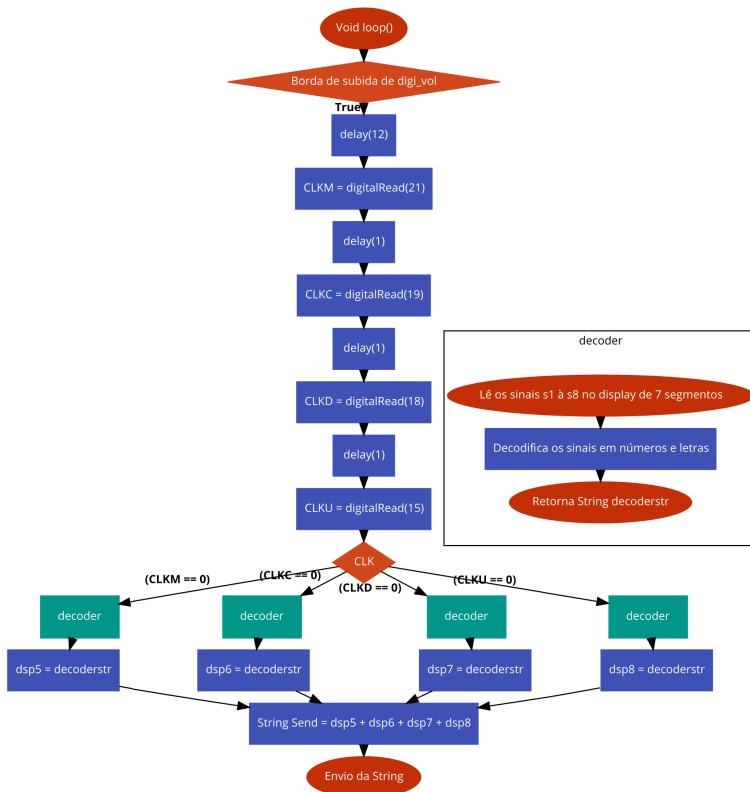


Figura 23 – Fluxograma do processamento do sinal do conjunto inferior de displays com a temporização correta.

pode fazer com que a decodificação fique prejudicada.

Devido a problemas de projeto da bomba, ao iniciar a infusão a decodificação dos sinais falha algumas vezes e, verificando os tempos de atraso, percebeu-se que eles foram modificados, conforme mostrado na figura 22.

Porém, não há grandes prejuízos com isso, uma vez que a única informação que é atualizada no display durante a infusão é o volume total infundido (informação disponível no conjunto de displays superior) e esse valor pode facilmente ser calculado pelo tempo de infusão e a vazão da bomba. Portanto, optou-se por não decodificar os sinais presentes no conjunto de displays superior.

3.2.2.1 Decodificador

Cada display de sete segmentos pode ser visto como um conjunto de 8 leds, ligados numa determinada sequência, mostram o caractere desejado.

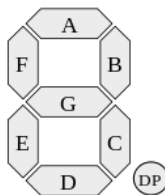


Figura 24 – Esquema de segmentos do display
FONTE: (PICUINO, 2018)

Verificando quais caracteres a bomba de infusão possui, criou-se uma tabela para decodificar os sinais desejados. É importante ressaltar que o display utilizado é anodo comum, ou seja, um nível lógico "0" liga um segmento e um nível lógico "1" desliga o segmento.

No código do projeto, a função "decoder" foi criada e, quando chamada, os sinais S1 a S8 são lidos e guardados num vetor, formando um código binário de 8 bits. Esse código binário é único para cada caractere. Assim, foi criado um "switch case" para cada código binário como mostra abaixo:

```

case 00000011:
decoderstr = '0'; //0
break;

case 00000010:
decoderstr = "0."; //0.
break;
```

A função atualiza a string "decoderstr" com o caractere identificado.

Tabela 1 – Decodificação os sinais dos displays de sete segmentos.

| DECODIFICADOR DISPLAY 7 SEGMENTOS | | | | | | | | |
|--------------------------------------|---|---|---|---|---|---|---|----|
| Caractere | a | b | c | d | e | f | g | dp |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 4 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 6 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 7 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 9 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0. | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1. | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 2. | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 3. | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 4. | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 5. | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 6. | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7. | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 8. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9. | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| "Vazio" | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| A | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| b | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| C | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| d | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| E | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| h | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| o | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| P | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| r | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| t | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| u | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| V | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| L | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |

4 INTEGRAÇÃO ENTRE ESP32 E MYSQL

Um dos objetivos do projeto é criar um histórico de infusões realizadas com determinada bomba e, principalmente, impedir que qualquer usuário possa alterar esses dados. O uso de um banco de dados, então, faz-se necessário. O banco de dados escolhido para o desenvolvimento do projeto foi o MySQL.

O MySQL é um sistema de gerenciamento de banco de dados de código aberto que utiliza a linguagem SQL como interface e, hoje, pertence à empresa Oracle (ORACLE, 2018).

A linguagem SQL é utilizada para “conversar” com o banco de dados. Com ela, pode-se criar tabelas, colunas, controle de usuários, e todas as funções necessárias para o seu gerenciamento completo.

Os softwares de gerenciamento de banco de dados geralmente são utilizados por meio da tela de comando do sistema operacional (prompt de comando, no caso do Windows), porém o MySQL disponibiliza gratuitamente o software MySQL Workbench, que permite realizar todo o gerenciamento de forma gráfica, facilitando muito o aprendizado de novos usuários.

4.1 CRIAÇÃO DO BANCO DE DADOS NO MYSQL

Na central de monitoramento, todas as bombas de infusão com o sistema irão se conectar ao mesmo servidor de banco de dados, de forma a concentrar toda a informação em um local só, pois não há a necessidade de criar um servidor para cada bomba em que o tráfego de dados não é tão intenso.

Cada bomba de infusão possuirá um banco de dados, também chamado de “schema”, único dentro do servidor, que será identificado pelo número de série da bomba.

Dentro do banco de dados da bomba de infusão selecionada, haverá duas tabelas: “led_status”, mostrada na imagem 26 e “display_status”, mostrada na imagem 27.

As colunas das duas tabelas correspondem às variáveis criadas na seção anterior, na interface entre bomba de infusão e ESP32. Porém, nas duas tabelas existe uma coluna a mais: “datetime”. Essa coluna é preenchida automaticamente com a data e a hora no momento em que um dado for recebido.

Percebe-se que na tabela “display_status” há apenas três colu-

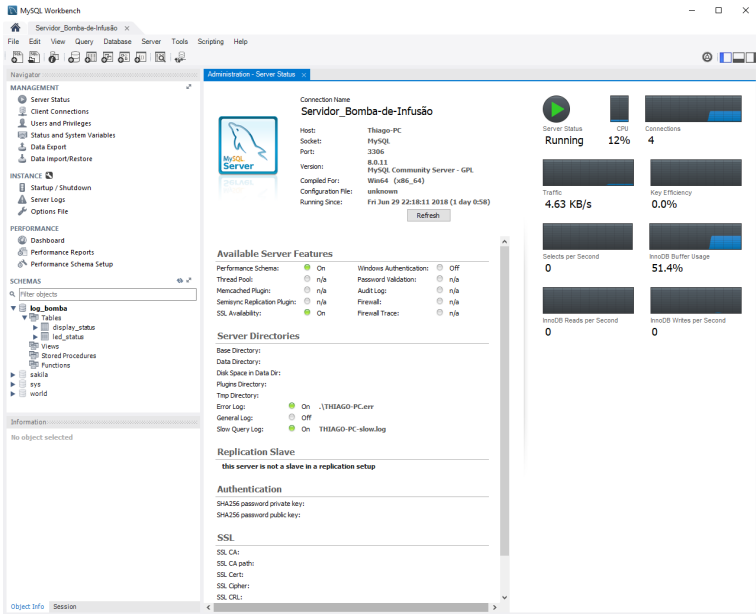


Figura 25 – Tela inicial do MySQL Workbench do gerenciador de banco de dados MySQL.

nas: "display_volume", "display_tempo" e "display_vazão". Essas três colunas correspondem às informações do conjunto de displays inferior da bomba de infusão pois, conforme dito anteriormente, decidiu-se não decodificar o conjunto superior de displays.

4.2 COMUNICAÇÃO ENTRE ESP32 E MYSQL

Após criar a interface entre bomba de infusão e ESP32 e criar o banco de dados, deve-se realizar a comunicação entre ESP32 e MySQL via Wi-Fi, como proposto inicialmente.

O MySQL disponibiliza em seu site diversas ferramentas para realizar a conexão do banco de dados com outras plataformas como Java, HTML, PHP, etc. Porém não existe nenhuma ferramenta oficial que possibilite a comunicação entre microcontroladores que utilizam a Arduino-IDE e o banco de dados.

Em buscas por soluções alternativas, encontrou-se o MySQL

| datetime_led | iniciar | parar | purgar | vol_lim | ten_lim | vazao | alarme_oclusao | alarme_vazao_livre | alarme_ar_na_linha | ar_na_linha_off | bat_low | bat | ar_na_linha_on | rede_on | |
|---------------------|---------|-------|--------|---------|---------|-------|----------------|--------------------|--------------------|-----------------|---------|-----|----------------|---------|---|
| 2018-06-17 03:42:43 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | |
| 2018-06-17 03:43:01 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | |
| 2018-06-17 03:43:06 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | |
| 2018-06-17 03:43:52 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 2018-06-17 03:45:30 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 2018-06-17 03:47:00 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 2018-06-17 03:47:26 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 2018-06-17 03:49:19 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 2018-06-17 03:49:33 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 2018-06-17 03:49:44 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

Figura 26 – Tabela “led_status” com suas respectivas colunas.

| datetime_display | display_volume | display_tempo | display_vazao |
|---------------------|----------------|---------------|---------------|
| 2018-06-17 03:43:08 | 354.5 | 47.15 | 7.5 |
| 2018-06-17 03:43:53 | 354.5 | 47.15 | 7.5 |
| 2018-06-17 03:49:19 | 354 | 0 | 0 |
| 2018-06-17 03:49:20 | 354.5 | 0 | 0 |
| 2018-06-17 03:49:33 | 354.5 | 47.15 | 0 |
| 2018-06-17 03:49:44 | 354.5 | 47.15 | 7.15 |
| 2018-06-17 03:49:45 | 354.5 | 47.15 | 7.5 |
| 2018-06-17 03:52:43 | 354.5 | 47.15 | 17.5 |
| 2018-06-17 04:02:34 | 354.5 | 47.15 | 17 |
| 2018-06-17 04:02:35 | 354.5 | 47.15 | 17.5 |
| 2018-06-17 04:07:35 | 354.5 | 47.15 | 117.5 |
| 2018-06-17 04:17:51 | 354.5 | 117.5 | 117.5 |
| 2018-06-17 04:24:14 | 354.5 | 117.5 | 117.5 |
| 2018-06-17 04:24:31 | 354.6 | 117.5 | 117.5 |

Figura 27 – Tabela “display_status” com suas respectivas colunas.

Connector/Arduino, biblioteca para Arduino desenvolvida pelo Prof. Chuck Bell (2016), membro da equipe de desenvolvimento do MySQL. A sua única limitação é que o microcontrolador em questão tenha ao menos 32 kB de memória.

De forma simplificada, o MySQL Connector/Arduino permite o envio de comandos ao MySQL como se estivesse tendo acesso direto ao banco de dados. Desta forma, basta criar uma frase com o comando desejado.

No projeto em questão, necessita-se ter acesso ao MySQL em dois momentos: para gravar dados na tabela “led_status” e para gravar dados na tabela “display_status”. Um exemplo simples de inserção de dados é:

```
String display_status = "INSERT INTO log_bomba.display_status
    (display_volume, display_tempo, display_vazao) VALUES("; //
    String com o texto inicial
String display = dsp5 + dsp6 + dsp7 + dsp8; //String com os
    caracteres decodificados nos displays de sete segmentos
```

```
String complete_display = display_status + display; //Concatena
    as duas strings
complete_display += ");"; //Concatena o parentesis e
    ponto-e-virgula ao final

int str_len = complete_display.length() + 1; //Calcula o tamanho
    da String +1
char char_array[str_len]; //Criar um array de caracteres do
    tamanho da string (Buffer)
complete_display.toCharArray(char_array, str_len); //Converte a
    String em Char

//Fazendo o envio da string para o MySQL
MySQL_Cursor *cursor = new MySQL_Cursor(&conn);
cursor->execute(char_array);
delete cursor;
```

5 CRIAÇÃO DO SOFTWARE PARA WINDOWS

O objetivo final do projeto é mostrar as informações da bomba de infusão em uma tela de computador de forma agradável ao usuário final. Conforme citado anteriormente, o MySQL disponibiliza diversas ferramentas para realizar a comunicação entre seu banco de dados e outros ambientes que utilizem outras linguagens de programação como PHP, C++, Python, etc.

O MySQL também disponibiliza ferramentas para comunicação com as plataformas Java e .NET Framework.

5.1 VISUAL STUDIO E .NET FRAMEWORK



Figura 28 – Windows Visual Studio.

FONTE: (MICROSOFT, 2018)

O Visual Studio é um ambiente de desenvolvimento da empresa Microsoft voltado para a criação de softwares especialmente dedicado ao .NET Framework e às linguagens Visual Basic, C, C++, C# e J#.

A iniciativa da Microsoft foi criar um ambiente de desenvolvimento capaz de oferecer suporte à criação de softwares para Windows, Mac e Linux, além de estrutura para desenvolvimento web e aplicações *mobile* para Android e iOS.

Com o intuito de gerar uma aplicação para o sistema operacional Windows, optou-se por utilizar o .NET framework, principalmente devido ao grande suporte e informações disponibilizadas pela empresa.

O .NET Framework, também criado pela Microsoft, é uma plataforma de desenvolvimento e execução de sistemas e aplicações. Sua principal vantagem é ser multiplataforma, em que um código .NET pode funcionar corretamente em qualquer sistema que possua um fra-

mework da plataforma. De forma simplificada, o código não é escrito para um sistema específico, e sim para a plataforma .NET utilizando diversas linguagens de programação disponíveis. A estrutura do Visual Studio utilizando o .NET Framework pode ser vista na imagem 29.

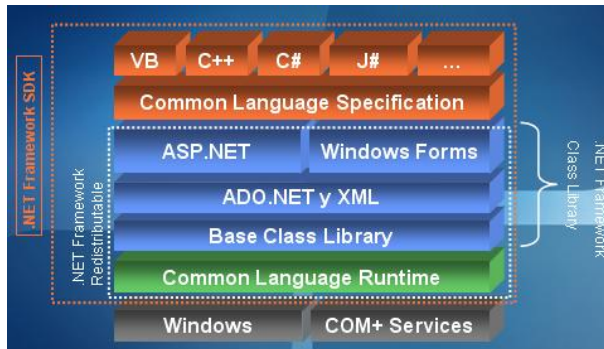


Figura 29 – Arquitetura do .NET Framework.
 FONTE: (AMAGUAYO, 2008)

Desta forma, a linguagem escolhida foi a C#, devido ao fato de ter sido desenvolvida como parte da plataforma .NET e, assim, possuir um suporte melhor para o desenvolvimento da aplicação. A C# é uma linguagem de programação orientada a objetos inspirada no C++ e fortemente inspirada em linguagens como Object Pascal e Java.

5.2 DESENVOLVIMENTO DA APLICAÇÃO

De forma simplificada, o desenvolvimento da aplicação se dividiu em duas partes: criação do design e programação das janelas e ícones criados no design. A tela inicial do programa pode ser vista na figura 30.

Para a criação do design, levou-se em conta todas as informações obtidas da bomba de infusão e tentou-se criar uma interface limpa e objetiva, como mostrado na figura 31.

A aplicação pode ser aberta diversas vezes de forma que em cada janela aberta pode ser configurada uma nova bomba de infusão inserindo apenas o número de série (Na figura 31 o número de série é representado pelo nome “log_bomba”).

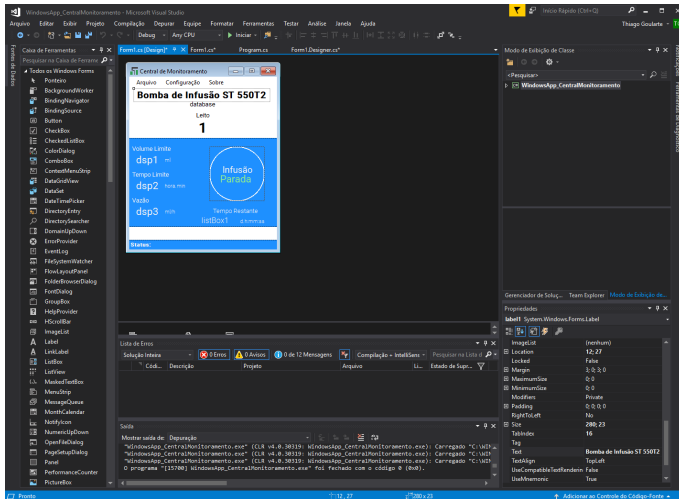


Figura 30 – Tela principal do ambiente de desenvolvimento do Visual Studio.



Figura 31 – Interface gráfica da aplicação criada.

A aplicação busca os dados diretamente do banco de dados e os atualiza nos seus respectivos locais. Por exemplo, se a infusão estiver parada, irá aparecer a palavra “Parada” em vermelho, mas se a infusão for iniciada, aparecerá a palavra “Iniciada” na cor verde no mesmo local. Pode-se ver um exemplo na figura 32.

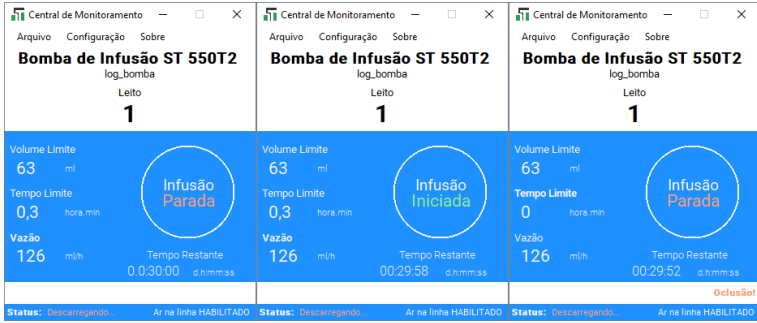


Figura 32 – Sequência de execução e parada de infusão.

O tempo restante mostrado é calculado realizando o quociente dos valores de volume limite e vazão obtidos e, ao iniciar a infusão, esse tempo é decrementado a cada segundo até chegar a zero.

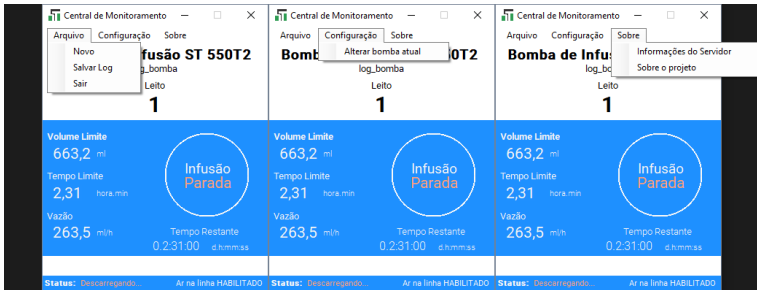


Figura 33 – Menus do software desenvolvido.

É importante mencionar que a aba de configuração foi adicionada, mas ainda não foi implementada. Desta forma, por enquanto ainda não é possível alterar a bomba de infusão utilizada.

6 PROTÓTIPO FINAL

Apesar do sucesso na integração entre Bomba de Infusão, ESP32, MySQL e aplicação Windows o projeto ainda não pôde ser finalizado devido ao fato de as conexões terem sido feitas de forma provisória, havendo a necessidade de desenvolver uma interface física mais robusta e confiável.

Utilizando o Software DipTrace, foi criado o esquemático indicado na figura 17 e, por fim, o layout da placa de circuito impresso. O layout foi feito inteiramente a mão, sem utilizar ferramentas de autorroteamento, pois há uma limitação muito grande de espaço e uma grande quantidade de conexões. O layout pode ser visto na figura 34.

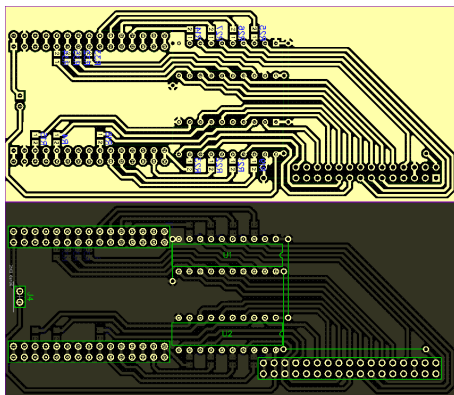


Figura 34 – Layout da placa de circuito impresso.

Assim, a placa de circuito impresso foi fabricada conforme mostrado nas imagens 35 e 36.

O layout foi desenvolvido de forma a não utilizar fios para fazer a conexão entre ESP32 e Bomba de Infusão, bastando conectá-la diretamente no barramento de sinais da placa traseira. É interessante notar que a escolha dos pinos de entrada do ESP32 foi feita para otimizar o roteamento das trilhas.

Desta forma, a placa de circuito impresso foi conectada no seu respectivo local e a bomba de infusão foi devidamente fechada.

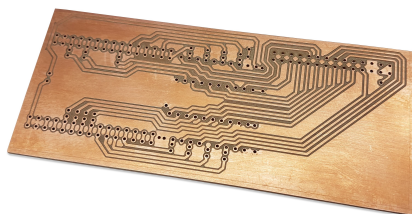


Figura 35 – Trilhas de cobre da placa de circuito impresso.

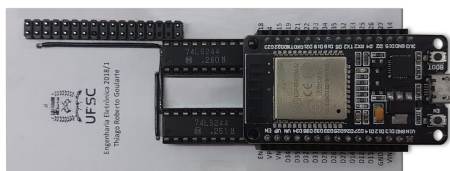


Figura 36 – Parte frontal da placa.

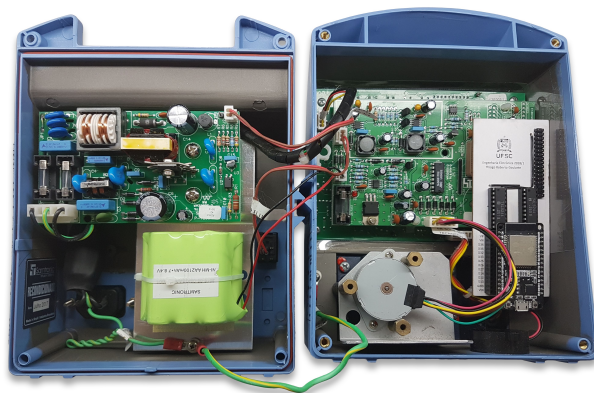


Figura 37 – Bomba de infusão aberta com a placa conectada.

7 CONSIDERAÇÕES FINAIS

O projeto possui um objetivo final bem definido: a criação de um sistema de monitoramento para a bomba de infusão ST 550T2. Apesar de o objetivo final ser simples, sua execução foi desafiadora, necessitando conhecimentos em análise de circuitos, sistemas digitais, sistemas embarcados, programação orientada a objetos e até mesmo layout de placa de circuito impresso.

O protótipo desenvolvido atendeu às expectativas iniciais, conseguindo realizar a tarefa para o qual foi desenvolvido: monitorar uma bomba de infusão de forma remota.

Ainda assim, existem alguns problemas e funções não implementadas no sistema:

- Ainda não é possível alterar a rede Wi-Fi conectada de forma remota, precisando fazer o upload do firmware diretamente no microcontrolador com o novo login e senha;
- No software desenvolvido ainda não é possível alterar a bomba escolhida no banco de dados, precisando recompilar o programa para tal;
- A definição do volume total infundido ainda não foi implementada no software;
- Ao pausar a infusão devido a um alarme, a bomba não zera o contador de tempo de infusão, podendo continuar de onde parou. Porém, o software zera o relógio cada vez que a infusão é paralisada.
- Ainda existe um pequeno atraso na atualização de informações do display, algo que não acontece com os leds.

Por se tratar de um produto que já está no mercado e ser necessária a modificação física em uma das placas, a implementação deste sistema pode ser dificultada, porém ele mostra sua capacidade, podendo servir como base para implementações em futuros equipamentos.

Outro ponto importante é o fato da bomba já ter sido certificada quanto à compatibilidade eletromagnética. No manual da fabricante é mostrado que a mesma é integrante do Grupo 1 nos testes de emissões eletromagnéticas CISPR 11, ou seja, ela possui apenas frequências de RF internamente (principalmente da fonte de alimentação chaveada).

Sendo assim, teóricamente, não causa interferência eletromagnética em outros equipamentos.

Porém, ao adicionar um microcontrolador com comunicação Wi-Fi na bomba de infusão, esta certificação deve ser feita novamente e, assim, verificando se ela ainda pode ser utilizada em ambiente hospitalar.

REFERÊNCIAS

- AMAGUAYO, R. *.Net Framework Presentation*. Novembro 2008. <<https://www.slideshare.net/robertojose23/net-framework-presentation>>. Acessado em 30 jun. 2018.
- ARDUINO. *detachInterrupt()*. <<https://www.arduino.cc/reference/en/language/functions/external-interrupts/detachinterrupt/>>. Acessado em 02 jun. 2018.
- AUGUSTO. *Converter Data dd/mm/aaaa para Formato ISO 8601 (aaaa-mm-dd)*. <<https://pt.stackoverflow.com/questions/11065/converter-data-dd-mm-aaaa-para-formato-iso-8601-aaaa-mm-dd>>. Acessado em 25 jun. 2018.
- BELL, C. *New Release! MySQL Connector/Arduino 1.1*. Janeiro 2016. <<http://drcharlesbell.blogspot.com/2016/01/new-release-mysql-connectorarduino-110a.html>>. Acessado em 20 mai. 2018.
- CANC. *Reading a 10 pin 7 segment (2 digits) using Arduino uno*. <<http://forum.arduino.cc/index.php?topic=260419.0>>. Acessado em 01 mai. 2018.
- DEV MEDIA. *Índices na prática numa aplicação windows Form C#*. <<https://www.devmedia.com.br/indices-na-pratica-numa-aplicacao-windows-form-csharp/30386>>. Acessado em 11 jun. 2018.
- DO BIT AO BYTE. *ESP32: delay, vTaskDelay, vTaskDelayUntil, millis*. <<https://www.dobitaobyte.com.br/esp32-delay-vtaskdelay-vtaskdelayuntil-millis/>>. Acessado em 02 jun. 2018.
- EDUARDO, M. *Conversão de Dados no C#*. <<http://ilovecode.com.br/conversao-de-dados-no-c/>>. Acessado em 11 jun. 2018.
- EDUARDO, M. *Introdução sobre a Linguagem C#*. <<http://ilovecode.com.br/introducao-sobre-a-linguagem-c/>>. Acessado em 5 jun. 2018.

ESPRESSIF. *ESP32 Technical Reference Manual*. Shanghai, China, 2018.

EXPRESSIF. *AttachInterrupt on NodeMCU*. <<https://bbs.espressif.com/viewtopic.php?t=3273>>. Acessado em 10 mai. 2018.

FELIX, A. *MySQL - Como Selecionar uma Coluna Distinta*. <<https://blog.ffelix.eti.br/mysql-como-selecionar-uma-coluna-distinta/>>. Acessado em 10 jun. 2018.

GITHUB. *Mysql library*. <<https://github.com/esp8266/Arduino/issues/916>>. Acessado em 20 mai. 2018.

KHAN, M. A. *DataGridView with MySql Database in Windows Formst*. <<https://www.aspsnippets.com/Articles/Bind-Populate-DataGridView-with-MySQL-Database-in-Windows-Forms-WinForms-Application-using-C-and-VBNet.aspx>>. Acessado em 15 jun. 2018.

MICROSOFT. *Como adicionar e remover itens de um controle ComboBox, ListBox ou CheckedListBox*. <<https://docs.microsoft.com/pt-br/dotnet/framework/winforms/controls/add-and-remove-items-from-a-wf-combobox>>. Acessado em 11 jun. 2018.

MICROSOFT. *ListBox Class*. <<https://docs.microsoft.com/pt-br/dotnet/api/system.windows.forms.listbox?view=netframework-4.7.2>>. Acessado em 16 jun. 2018.

MICROSOFT. *Recuperando dados usando um DataReader*. <<https://docs.microsoft.com/pt-br/dotnet/framework/data/adonet/retrieving-data-using-a-datareader>>. Acessado em 15 jun. 2018.

MICROSOFT. 2018. <<http://netcoders.com.br/visual-studio-15-abertura-pastas/>>. Acessado em 02 jul. 2018.

MITCHELL, S. *MySQL Error 1064: You have an error in your SQL syntax*. <<https://www.inmotionhosting.com/support/website/database-troubleshooting/error-1064>>. Acessado em 20 mai. 2018.

MIWPE. [C#] *Thread - Atualizar DataGridView "automaticamente"*. <<https://social.msdn.microsoft.com/Forums/pt-BR/7954fe59-63d6-4088-93b1-80428e8481ca/c-thread-atualizar-datagridview-automaticamente?forum=clientept>>. Acessado em 16 jun. 2018.

MORAIS, J. *Manipulando os Registradores - ESP32*. <<https://portal.vidadesilicio.com.br/manipulando-os-registradores-esp32/>>. Acessado em 09 mai. 2018.

MYSQL. *MySQL Windows Forms Items*. <<https://dev.mysql.com/doc/visual-studio/en/visual-studio-project-items-forms.html>>. Acessado em 05 jun. 2018.

NEXPERIA. *Octal D-type flip-flop with reset 74HC273 Datasheet*. Nimega, Países Baixos, 2016.

ORACLE. 2018. <<https://www.mysql.com/>>. Acessado em 04 jun. 2018.

PICUINO. 2018. <<http://www.picuinio.com/es/arduprog/pc42-display1.html>>. Acessado em 02 jul. 2018.

RACHED, E. *Connect C# to MySQL*. <<https://www.codeproject.com/Articles/43438/Connect-C-to-MySQL>>. Acessado em 5 jun. 2018.

RODSTAR. *How to read an equipment's 7 segment display*. <<https://forum.allaboutcircuits.com/threads/how-to-read-an-equipments-7-segment-display.128959/>>. Acessado em 01 mai. 2018.

SAMTRONIC. *Manual do usuário Bomba de infusão peristáltica ST550T2*. São Paulo, Brasil, 2007.

SANTOS, N. *ESP32 Arduino: External interrupts*. Setembro 2017. <<https://techtutorialsx.com/2017/09/30/esp32-arduino-external-interrupts/>>. Acessado em 1 jun. 2018.

SMITH, K. C.; CEDRA, A. S. *Microelectronic Circuits: Theory And Applications*. Oxford, Reino Unido: OXFORD UNIVERSITY, 2009. 7, 69–70,1204–1208 p.

SPARKFUN. *Logic Converter*. 2018. <<https://cdn.sparkfun.com/assets/f/3/3/4/4/526842ae757b7f1b128b456f.png>>. Acessado em 11 Jul. 2018.

SPICE, S. *How to Display column from MySql into windows form.* <<https://stackoverflow.com/questions/36366649/how-to-display-column-from-mysql-into-windows-form-text-box-on-visual-studio-c>>. Acessado em 10 jun. 2018.

STACK EXCHANGE. *Arduino directly to MySQL server using the URL.* <<https://arduino.stackexchange.com/questions/16347/how-do-i-connect-arduino-directly-to-mysql-server-using-the-url>>. Acessado em 10 mai. 2018.

SUNLED. *Single Digit Numeric Display XDMR08A Datasheet.* Califórnia, Estados Unidos, 2014.

TAMBI, Y. *Seven Segment Multiplexing.* <<http://maxembedded.com/2013/01/seven-segment-multiplexing/>>. Acessado em 01 mai. 2018.

TARGET TRUST. *Consultando Dados com SQL (Comando SELECT).* <<https://targettrust.com.br/blog/comando-select/>>. Acessado em 10 mai. 2018.

VIANA, P. C. *Contador regressivo em C#.* <<https://social.msdn.microsoft.com/Forums/pt-BR/c94b1fbd-acb6-4405-b941-7f5599a22261/contador-regressivo-em-c?forum=vssharppt>>. Acessado em 25 jun. 2018.